Bejuri *et al. J Big Data (2017) 4:33*
DOI 10.1186/s40537-017-0094-3

Journal of Big Data

RESEARCH Open Access

Adaptive memory-based single distribution resampling for particle filter

Wan Mohd Yaakob Wan Bejuri[1,2*], Mohd Murtadha Mohamad[1], Raja Zahilah Raja Mohd Radzi[1], Mazleena Salleh[1] and Ahmad Fadhil Yusof[1]

*Correspondence: mr.wanmohdyaakob.my@ieee.org
[1] Faculty of Computing, Universiti Teknologi Malaysia, Johor Bahru, Johor, Malaysia
Full list of author information is available at the end of the article

Abstract

The restrictions that are related to using single distribution resampling for some specific computing devices' memory gives developers several difficulties as a result of the increased effort and time needed for the development of a particle filter. Thus, one needs a new sequential resampling algorithm that is flexible enough to allow it to be used with various computing devices. Therefore, this paper formulated a new single distribution resampling called the adaptive memory size-based single distribution resampling (AMSSDR). This resampling method integrates traditional variation resampling and traditional resampling in one architecture. The algorithm changes the resampling algorithm using the memory in a computing device. This helps the developer formulate a particle filter without over considering the computing devices' memory utilisation during the development of different particle filters. At the start of the operational process, it uses the AMSSDR selector to choose an appropriate resampling algorithm (for example, rounding copy resampling or systematic resampling), based on the current computing devices' physical memory. If one chooses systematic resampling, the resampling will sample every particle for every cycle. On the other hand, if it chooses the rounding copy resampling, the resampling will sample more than one of each cycle's particle. This illustrates that the method (AMSSDR) being proposed is capable of switching resampling algorithms based on various physical memory requirements. The aim of the authors is to extend this research in the future by applying their proposed method in various emerging applications such as real-time locator systems or medical applications.

Keywords: Particle filter, Resampling, Sequential implementation, Memory consumption

Introduction

Over the past two decades, particle filtering has emerged as a procedure for sequential signal processing (Refs. [1–4] presents the review). Particle filter has also become popular because it is capable of processing observations signified by nonlinear state-space models. In such models, the noises can be non-Gaussian. Several fields have adopted this methodology including: finance [5–8], wireless communications [9–12], geophysical systems [13–17], navigation and tracking [18–20], control [21–25], and robotics [26–31]. Generally, this methodology can approximate state density $p(x_k)$ using a range of random particles that have related nonnegative weights:

$$p(x_k) \approx \sum_{k=1}^{N_k} w_{k,i}\delta_{x_{k,i}}(x_k), s.t. \sum_{k=1}^{N_k} w_{k,i} = 1,\ w_{k,i} \geq 0 \tag{1}$$

where $\{x_{k,i}, w_{k,i}\}_{i=1,2,\ldots,N_k}$ represent the particles' states and weights, N_k represents the total amount of particles at time k, and $\delta_x(\cdot)$ represents the delta-Dirac mass found in x. The systematic important sampling principle was used to choose the weights, for example:

$$w_k \propto w_{k-1}\frac{p(y_k|x_k)p(x_k|x_{k-1})}{p(x_k|x_{k:k-1},y_k)} \tag{2}$$

where y_k represents the measurement for time k, and $q(\cdot)$ represents the proposal importance density. It has been established that after a small amount of iterations in the process of particle propagation, the weight will be exclusively concentrated on a few particles. Alternatively, the weight of the majority will be negligible, resulting in sample degeneracy [5]. This problem can be solved by resampling, which gets rid of particles with small weights and focuses on heavier particles. Resampling algorithms come in various kinds (as elaborated below). However, the simplest is known as the single distribution resampling. This is thought of as the simpler of the two kinds of resampling algorithms: traditional variation resampling and traditional resampling. These algorithms have significant differences when it comes to their sampling strategy. For traditional resampling, each j cycle only has one sampling, while the traditional sampling performs more than one sampling for every j cycle. Thus, compared to traditional variant resampling, traditional resampling needs less memory for every k cycle. Because of the variations in computing systems (from handheld embedded systems to enormous supercomputers), there is a rising need for a new sequential resampling algorithm that needs less memory while operating within a low memory-based computer system while needing additional memory if it operates in a high memory-based computer system.

Therefore, this paper proposes the development of a new sequential resampling algorithm that can be used for various memory platforms with the use of integrated sampling of traditional variant resampling and traditional resampling. The organisation of this paper is as follows: "Background to the single distribution resampling particle filter" presents a review of the related papers. "Problem formulation" provides an outline of the problem's formulation. "Objective" presents the important research hypotheses and the study's major objectives. "Adaptive memory size-based single distribution resampling (AMSSDR)" studies the detailed design used for the adaptive memory size based resampling (AMSSDR). "Experimental result" presents the result of this proposed method. Lastly, "Conclusions and future implementations" discusses the future implications of this study.

Background to the single distribution resampling particle filter

This study was introduced in the previous section. For this current section, the background of the single distribution resampling particle filter is discussed in three sub sections: (1) particle filter; (2) resampling; and (3) single distribution resampling. "The

particle filter" discusses the general concept behind the particle filter. "Resampling" provides a discussion of the basic idea of resampling, which is one of particle filter's components. "Single distribution resampling" provides a discussion of the basic idea of single distribution resampling based on the various categories of single distribution resampling.

The particle filter

This part will provide a discussion on the general concept behind a particle filter. It begins with a short review about particle filter as well as the notation's introduction. The state-space model is described in the following manner:

$$x_t = g(x_{t-1}, u_t), \tag{3}$$

$$y_t = h(x_t, v_t), \tag{4}$$

Where t represents a time index and $t = 1, 2, \ldots$; $x_t \in R^{dx}$ represents the state of the hidden model (for instance, not observed); $y_t \in R^{dy}$ represents the observation; $u_t \in R^{du}$ and $v_t \in R^{dv}$ represent the white noises that are not dependent on each other; and $g: R^{dx} \times R^{du} \to R^{dx}$ and $h: R^{dx} \times R^{dv} \to R^{dy}$ represent known functions. Alternatively, these equations can be represented by the state's probability distributions, $p(x_t|x_{t-1})$, and by the observation, $p(y_t|x_t)$, which one can gather from (1) and (2) and the u_t and v_t or probability distributions, v_t, respectively. The focus is on nonlinear models when the noises observed in (1) and (2) need not necessarily be Gaussian. The particle filter aims to sequentially estimate the state's distributions, including the predictive distribution $p(x_t|y_{1:t-1})$, the filtering distribution $p(x_t|y_{1:t})$, or the smoothing distribution, $p(x_t|y_{1:T})$, where $t < T$. The focus of this part is on the filtering distribution. The expression of this distribution can be based on the filtering distribution during time instant $t-1, p(x_{t-1}|y_{1:t-1})$—for instance, in a recursive form using;

$$p(x_t|y_{1:t}) \propto \int p(y_t|x_t)p(x_t|x_{t-1})p(x_{t-1}|y_{1:t-1})dx_{t-1}, \tag{5}$$

where $\propto$ implies being 'proportional to.' Except in rare cases, one cannot analytically implement this update. Thus, the authors have to approximate, while emphasising that with particle filter, the fundamental approximation is a representation of the continuous distributions by discrete random measures that are made up of particles $x_t^{(m)}$, which can be probable values of the unknown weights $w_t^{(m)}$ and state x_t and were given to the particles. One can approximate the distribution $p(x_{t-1}|y_{1:t-1})$ using a random measure having the form $X_{t-1} = \left\{x_{t-1}^{(m)}, w_{t-1}^{(m)}\right\}_{m=1}^{M}$, where M represents the amount of particles, like in the example below:

$$p(x_{t-1}|y_{1:t-1}) \approx \sum_{m=1}^{M} w_{t-1}^{(m)} \delta\left(x_{t-1} - x_{t-1}^{(m)}\right) \tag{6}$$

where $\delta(\cdot)$ represents the Dirac delta impulse with all the weights adding up to one. Given this approximation, one can readily solve the integral in (3) and express it as follows:

$$p(x_t|y_{1:t}) \dot{\propto} p(y_t|x_t) \sum_{m=1}^{M} w_{t-1}^{(m)} p\left(x_t|x_{t-1}^{(m)}\right) \tag{7}$$

where $\dot{\propto}$ signifies 'approximate proportionality'. The final expression is a demonstration of how the filtering distribution's approximation x_t can be recursively obtained overtime. During time instant $t-1$, the development of x_t begins by the generation of particles $x_t^{(m)}$, which represent $p(x_t|x_{t-1})$. This particle filter step is called particle propagation, since particle $x_{t-1}^{(m)}$ moves forward through time and is considered the parent of $x_t^{(m)}$. The importance sampling concept is used for weight computation and particle propagation [15]. Ideally, each of the propagated particles has to be taken from $p(x_t|y_{1:t})$ in order to obtain equal weights. However, this is not feasible for most cases, therefore necessitating the utilisation of an instrumental function $\pi(x_t)$ (as in [32]), with the $p(x_t|x_{t-1})$ function. The particle filter's second basic step is calculating particle weights. To find a right inference based on the generated particles, the theory shows how the generated particles from $\pi(x_t)$, are different from $p(x_t|y_{1:t})$, and therefore need to be weighted [33–35]. When working in mild conditions, one can demonstrate how these weights can be computed recursively based on:

$$w_t^{(m)} \propto w_{t-1}^{(m)} \frac{p\left(y_t|x_t^{(m)}\right) p\left(x_t^{(m)}|x_{t-1}^{(m)}\right)}{\pi\left(x_t^{(m)}\right)}. \tag{8}$$

It has often been recognised that the calculation of the expression that is found on the right side of the proportionality sign is succeeded by weight normalisation (for example, they should add up to one). Ideally, the particles' weights have to be the same. However, it is also extremely undesirable if each particle's weights are equal to zero, or if one (or some) particles make up most of the weight and the remaining particle weights become negligible. This is often referred to as degeneracy. It has been proven to occur when the particle filter is designed using only the two previously mentioned steps [36–39]. As the observation processing progresses, the weight variance increases until it gets to a point where the random measure resembles a very poor filtering distribution approximation. Thus, the need to have a third step, known as resampling, has arisen.

Resampling

This section provides a discussion of the basic idea of resampling, which is one of the particle filter's components. Resampling aims to prevent the propagated particles' degeneracy by altering the random measure X_t to $\tilde{X}_t$ and enhancing the state space examination at $t+1$. While addressing degeneracy during resampling, it is also important for the random measure to approximate the original distribution as precisely as possible so that bias in the estimates can be prevented [40–43]. Although the $\tilde{X}_t$ approximation closely resembles that of X_t, the set of $\tilde{X}_t$ particles have important variations from that of X_t. Resampling makes sure that X_t particles that are heavier will have greater tendency to dominate $\tilde{X}_t$ compared to lighter particles. This leads to the creation of additional new particles in the region having heavier particles during the subsequent time step. This results into exploration improvements after resampling. Furthermore, the focus of

exploration moves to the portions of space that possess large probability masses. Due to resampling, the particles propagated from $\tilde{X}_t$ will possess less discriminate weights compared to the propagation using the X_t particles. This concept is intuitive and has significant theoretical and practical implications. Formally, resampling refers to a process that takes samples from the original random measure $X_t = \left\{x_t^{(m)}, w_t^{(m)}\right\}_{m=1}^{M}$ so that it can generate a new random measure $\tilde{X}_t = \left\{\tilde{x}_t^{(n)}, \tilde{w}_t^{(n)}\right\}_{n=1}^{N}$. For the particles of X_t, the random measure is then replaced with $\tilde{X}_t$. Some of the particles are replicated during this process. The replicated particles are often the heaviest ones. The particles are mainly used to propagate new particles, and are thus considered the parents of $x_{t+1}^{(m)}$.

One should note that in order to approximate $p(x_t|y_{1:t})$, using X_t is more effective than $\tilde{X}_t$. It was also observed that the amount of resampled particles $N$is not equal to the amount of propagated particles all the time. Traditional resampling methods help maintain their value, and, generally, $M = N$. Lastly, for most resampling methods, the particle weights after resampling become equal. However, resampling may produce undesired effects, such as sample impoverishment. During resampling, it is likely for low weighted particles to be removed. Thus, the diversity of the particles is reduced [32, 44–48]. For instance, if a small amount of particles of X_t has the greatest weights, numerous resampled particles will end up being the same (there will be lesser distinct particles in $\tilde{X}_t$). The next effect is on the particle filter's speed of implementation. Often, particle filter is used to process signals when there is a need for the real-time processing of observations. An effective solution is to parallelise the particle filter. Later, it will be demonstrated how the process of parallelising the resampling can be a challenging one. Resampling's undesired effects have encouraged researchers to develop advanced resampling methods. These methods have a vast range of features, which include a variable amount of particles, the avoidance of rejecting low weighted particles, the removal of the restriction of the resamples needing equal weights, and the introduction of parallel frameworks that can be used during resampling. Several decisions are needed when conducting resampling, including: specifying the sampling strategy; choosing the distribution for resampling, determining the resampled size; and choosing the resampling frequency.

Single distribution resampling

This section will provide a discussion of the basic concept of single distribution resampling. Its focus will be on the various kinds of single distribution resampling that has been produced. Currently, single distribution resampling has two general categories: traditional variation resampling and traditional resampling. One can observe significant differences between these two methods. For instance, traditional resampling functions possess a single sample for every j cycle, while traditional variation resampling possesses a different function for each (demonstrated in Fig. 1). Some algorithms that are classified as traditional resampling include systematic, multinomial, residual, and stratified. On the other hand, traditional variation resampling has three algorithms: branch kill, residual systematic and rounding copy.

The categorisation for traditional resampling is considered a basic algorithm that makes use of numerous particle filters. The first is called multinomial resampling and is seen as a basic approach. The main notion behind multinomial resampling [49] includes the generation of independent random N numbers, $u_t^{(m)}$ that are taken from a uniform

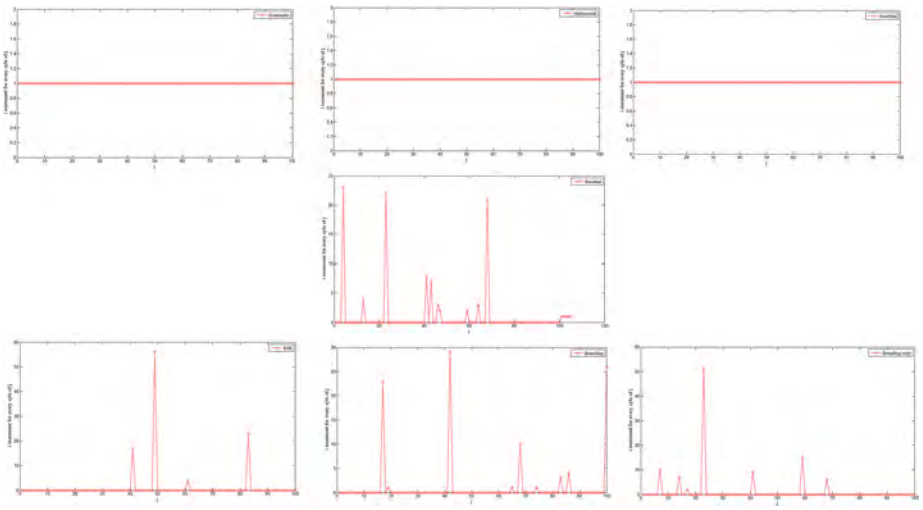

Fig. 1 Simulation illustration of sampling; the traditional resampling constant, sampling one for each j cycle; meanwhile, the traditional variation resampling varies sampling for each j cycle

distribution on (0, 1]. They are then applied during the selection of particles from the $x_t^{(m)}$. When conducting the nth particle selection, one chooses a particle $x_t^{(m)}$ if it satisfies the conditions presented below:

$$Q_t^{(m-1)} < u_t^{(n)} \leq Q_t^{(m)} \tag{9}$$

Where

$$Q_t^{(m)} = \sum_{k=1}^{m} w_t^{(k)}. \tag{10}$$

Thus, the likelihood of the chosen $x_t^{(m)}$ and $u_t^{(n)}$ existing in the interval that is bounded by the normalised weights' total sum of is quite the same. The second algorithm is called stratified resampling [50]. This algorithm divides the entire particle population into smaller populations known as strata. Furthermore, it is responsible for pre-partitioning the $(0, 1]$ interval to create N disjoint subintervals $\left(\frac{0,1}{N}\right] \cup \cdots \cup \left(1 - \frac{1}{n}, 1\right]$.

For each subinterval, there is an independent drawing of random numbers $\left\{u_t^{(n)}\right\}_{n=1}^{N}$ in the following manner;

$$u_t^{(n)} \sim U\left(\frac{n-1}{N}, \frac{n}{N}\right], \quad n = 1, 2, \ldots, N, \tag{11}$$

Thus, the bounding scheme, which depends on the sum of all the normalised weights, is utilised. The third algorithm is called systematic resampling [51].

Essentially, it also explores the idea of strata [50, 51], but in a different manner. Here, the $u_t^{(1)}$ is chosen from a uniform distribution that is found on $\left(\frac{0,1}{N}\right]$, while the rest of the u numbers are obtained deterministically as in the following;

$$u_t^{(1)} \sim U\left(\frac{0,1}{N}\right],$$

$$u_t^{(n)} = u_t^{(1)} + \frac{n-1}{N}, \quad n = 2, 3, \ldots, N. \tag{12}$$

The final algorithm categorised as traditional resampling is residual resampling [52]. Generally, it is contains two main steps. In the first step, there is a deterministic replication of the particles having a weight greater than $1/N$ while the second step involves the random sampling with the help of the remaining weights (called as residuals). Code 2 represents the deterministic replication, whereas $N_t^{(m)}$ describes the number of times the $x_t^{(m)}$ particle is replicated in this manner. In the residual resampling scheme, the mth particle can be resampled $N_t^{(m)} + R_t^{(m)}$ times, whereas, $N_t^{(m)}$ and $R_t^{(m)}$ refers to the replication number obtained from the earlier two steps, wherein, $N_t^{(m)} = Nw_t^{(m)}$. The overall number of the particles replicated in the step 1, can be estimated by $N_t = \sum_{m=1}^{M} N_t^{(m)}$, while in step 2, the number is calculated as $R_t = N - N_t$. The residual of all the weights can be determined using the following equation:

$$\hat{w}_t^{(m)} = w_t^{(m)} - \frac{N_t^{(m)}}{N} \tag{13}$$

For the second step, the particle is chosen in terms of their residual weights and using the aid of multinomial resampling (one can also use any other random sampling scheme), where the probability of choosing $x_t^{(m)}$ is in direct proportion to particle residual weight.

The previously mentioned algorithms are the most popular, and commonly used conventional algorithms. They are also called traditional resampling. However, many researchers have modified them based on their needs. This led to the birth of the second classification of single distribution resampling called traditional variation resampling. For this portion, the computationally dearer portion about the multinomial resampling within the residual resampling algorithm was omitted, and the implementation of the resampling was done in one single loop. This algorithm is called residual systematic resampling. This process collects the fractional contributions that each particle found in a search systematic contributes until a sample can be generated (in a manner that is the same to the collection idea that was implemented in the systematic resampling method). No added procedure for residuals is needed. Hence, one can achieve one iteration loop having a complexity of $O(N)$ order. If one can vary the particle size in example M at each time step, this can be achieved by having the particles present in parallel and in a single loop. However, it is possible if keeping the particle size constant is not required at every time step and the size can be made to vary. There is also another easy way to manage particles that are in one loop and are parallel. Previous studies have described two approaches—branch kill resampling [53] and rounding copy resampling [54]. In branch kill resampling, one can represent the replicated particle $x_t^{(m)}$ number as $N_t^{(m)} = |Nw_t^{(m)}|$, which has a probability of $1 - p$ or corresponding to $N_t^{(m)} = |Nw_t^{(m)}| + 1$ with a probability p, where in $p = N_t^{(m)} = |Nw_t^{(m)}|$. In rounding copy resampling, $N_t^{(m)}$ is used to represent the closest $Nw_t^{(m)}$ integer—for instance, when the $N_t^{(m)}$ is rounded. These two algorithms do not need any additional operations and they can also satisfy the unbiasedness condition, even if they do not produce varying sample sizes. It was observed that for the three algorithms called residual systematic, branch kill, and the rounding

copy, the amount of replications for the mth particle's higher and the lower limits are $N_t^{(m)} = \lfloor Nw_t^{(m)} \rfloor$, respectively. The next section will talk about the problem formulation that was introduced in this paper.

Problem formulation

The previous part talked about the basic idea behind single distribution resampling. The notion being discussed is focused on the categorisation and differences of single distribution resampling. This section will talk about the issues that this paper raises. Generally, for all computing systems (from embedded hand-held systems to large supercomputers), memory systems primarily dictate power consumption, application performance, and reliability. The rising data-intensive nature of advanced applications requires significantly increased levels of performance within the limits imposed by power and cost. However, this is beyond the ability of traditional memory technologies. For instance, every minute, more than 100 h of video are uploaded to YouTube [55] and 250,000 photos are uploaded on Facebook [56]. In a similar manner, it is anticipated that extreme scale systems in the future will have to handle several exabytes of data. However, utilising a hard disk to support restart or checkpoint in these systems can lower their performance by over half [57]. Generally, software that has a particle filter needs high quantities of particle optimisation processing—for instance, a process called resampling.

There are several kinds of resampling methods, and the basic resampling method is called the single distribution resampling. Furthermore, single distribution resampling is subdivided into two categories called: traditional variation resampling and traditional resampling. Using traditional resampling benefits computing devices that need a single sampling process for every j cycle (for instance, computing devices that only have low memory requirements). On the other hand, the utilisation of traditional variation resampling is beneficial for computing devices that require more than a single sampling process for every j cycle (for instance, computing devices that have a high memory requirement). Although computing devices that have high memory requirement do not

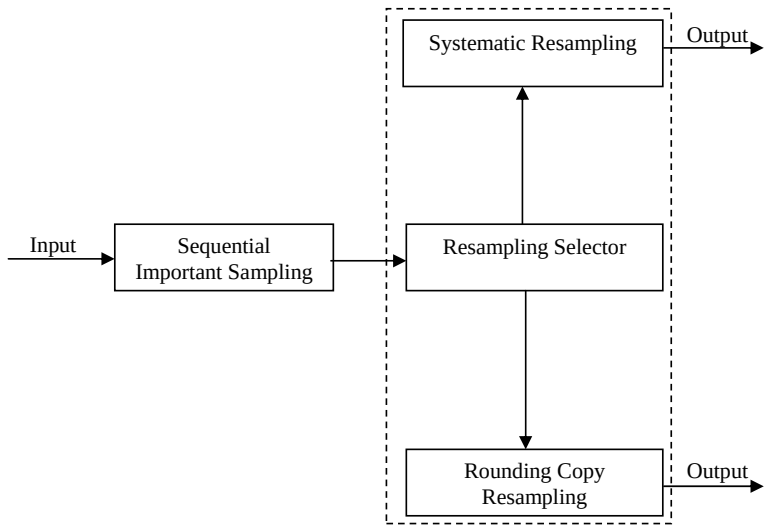

Fig. 2 Block diagram of AMSSDR (inside the dash line box) in single distribution resampling particle filter

normally have problems with memory consumption, traditional variation resampling is often thought to be more appropriate for usage in this environment because it is faster than traditional resampling [58].

By limiting the utilisation of single distribution resampling for particular computing devices, memory-related issues have proven to be difficult for the developer because they resulted into the need for more time and effort during particle filter development. This meant that a new sequential resampling algorithm was needed. This new algorithm must demand less memory when it is used in a low memory-based computer system and it must require additional memory when it is used in a high memory-based computer system. Thus, to address this issue, a new solution is needed, as elaborated in the next section. This could benefit the developer, since it could simply be applied in the particle filter of any computing device without having to take into account the issues of memory requirement. The next section will talk about the objective that was presented in this current paper.

Objective

The previous section talked about the issue that was introduced in this paper, specifically the utilisation of resampling in various memory computing devices. In this current section, the solution to this issue will be outlined. Therefore, the aim of this paper is to outline a new and credible method to conduct single distribution resampling, which is called the adaptive memory size based resampling. This new method is based on the integration of traditional variation resampling and traditional resampling in a resampling architecture. The algorithm is capable of switching the resampling algorithm based on computing device's memory. Applying this algorithm can help the developer develop a particle filter more easily without having to immediately consider a computing device's memory utilisation when different particle filter development processes are involved. The next section will talk about the design of the proposed algorithm that was presented in this paper.

Adaptive memory size-based single distribution resampling (AMSSDR)

The previous section talked about this paper's objective (the introduced solution based on the integration of traditional variation resampling and traditional resampling). This current section will talk about the proposed design of AMSSDR or the single distribution resampling. Figure 2 shows the AMSSDR's block diagram (inside the dash box) for the single distribution resampling particle filter. It can be seen in this figure that there are four major components; sequential important sampling; systematic important resampling (discussed in more detail in "Systematic resampling"); AMSSDR selector (discussed in more detail in "Adaptive memory size-based single distribution resampling selector"); and rounding copy resampling (discussed in more detail in "Rounding copy resampling"). Sequential sampling is utilised to produce the computation of the particle and weight, followed by AMSSDR (demonstrated in the dash box). The AMSSDR selector is implemented to choose an appropriate resampling algorithm (both for traditional variation resampling and traditional resampling), based on the current computing devices' physical memory. For this proposed algorithm, the systematic important resampling was chosen as traditional resampling while the rounding copy is used as the

traditional variation resampling. They were chosen because they were seen as the best approach within their group or resampling categorisation. If one chooses systematic resampling, the resampling will have to sample every particle for each *j* cycle. On the other hand, if one chooses the rounding copy resampling, the resampling will need to sample over one particle for every *j* cycle.

Adaptive memory size-based single distribution resampling selector

The previous section talked about the general AMSSDR. Its solution is made up of three main components: (1) resampling selector; (2) systematic important resampling; and (3) rounding copy resampling. This current section talks about the AMSSDR selector that was taken note of in the previous section. The resampling selector's main purpose is to alter the resampling operation between traditional variation resampling and traditional resampling, based on memory adaptation. This allows the operation of resampling in an optimum manner, based on the computing devices' physical memory requirements. Code 1 illustrates the pseudocode that was used for the resampling selector. Primarily, the pseudocode determines the total quantity of physical memory that is presently used by a computing device. If it is determined to be beyond 1536 MB, the resampling selector will select the resampling rounding. On the other hand, the resampling selector will select the systematic resampling algorithm to serve as its resampling operation. The next section talks about the operation systematic resampling.

Code 1: Adaptive Memory Size-based Single Distribution Resampling selector

```
PROGRAM: AMSR
GET Physical Memory;
mbmemory = Physical Memory/1024^2;
IF (mbmemory > 1536)
        THEN [indx] = resampleRoundingalgorithm(w, N);
       ELSE [indx] = resampleSystematicalgorithm(w, N);
END IF;
END
```

Systematic resampling

The preceding section talked about the AMSSDR selector's operation, which was utilised in the switching of the resampling operation between traditional variation resampling and traditional resampling, the basis of which is memory adaptation. Systematic resampling is the traditional resampling algorithm used, while rounding copy resampling is the traditional variation resampling algorithm implemented. This section will talk about the operation of systematic resampling (Code 2 can be used to refer to the pseudocode), which may be utilised the physical memory falls below 1.5 GB (1536 MB). First, it will make a sample $u_1 \sim U\left(\frac{0,1}{N}\right)$and give a definition to $u_i = u_1 + \frac{(i-1)}{N}$ for $i = 2,\ldots N$. Lastly, it utilises u_i to choose a particle that has an index i based on the multinomial distribution.

Code 2: Systematic resampling

$[\{\tilde{x}_t^{(n)}\}_{n=1}^{N}]$ = Resample $[\{x_t^{(m)}, w_t^{(m)}\}_{m=1}^{M}$,N]
$[\{Q_t^{(m)}\}_{m=1}^{M}]$ = CumulativeSum$[\{w_t^{(m)}\}_{m=1}^{M}]$
$n = 0$
$m = 1$
WHILE $(n \leq N)$
$u = u_0 + n/N$
WHILE $\left(Q_t^{(m)} < u\right)$
$m = m + 1$
END
$n = n + 1$
$\tilde{x}_t^{(n)} = x_t^{(m)}$
END

Rounding copy resampling

The earlier section talked about sequential importance resampling, which utilises the algorithm if a computing device's physical memory falls below 1.5 GB. This section talks about the rounding copy resampling (as demonstrated in Code 3), which is utilised if the physical memory exceeds 1.5 GB (1536 MB). It has been demonstrated that every particle is strictly resampled $n_i = [N \times W_i]$ times. Here, the symbol is used to represent the nearest integer that is around the contents. This creates a pure deterministic sampling algorithm without having to introduce randomness. For the rounding operation, $E[N \times W_i] = NW_i$ are under the unbiasedness condition. When both sides are summed, one obtains the following equation:

$$E\left(N^*\right) = N \tag{14}$$

Moreover, the scope of N* can be presented as follows:

Remark 1

$$max\left(N - \frac{M}{2}, 0\right) <\approx [N* \approx N + \frac{M}{2}).$$

Proof For the rounding operation on every particle, the following is obtained:

$$|N \times W_i - [N \times W_i]| <\approx 1/2 \tag{15}$$

where $<\approx$ is used to signify something infinitely close to and smaller. Conversely, one can determine the absolute value inequality as:

$$\left|\sum_{i=1}^{M}(N \times W_i) - \sum_{i=1}^{M}(N \times W_i)\right| \leq \sum_{i=1}^{M}|N \times W_i - [N \times W_i]| \tag{16}$$

Code 3: Rounding copy resampling

$[\{\tilde{x}_t^{(n)}\}_{n=1}^{N}]$ = Resample $[\{x_t^{(m)}, w_t^{(m)}\}_{m=1}^{M}, N_{ref}]$
FOR $m = 1:M$
$N_t^{(m)} = Round(N_{ref} \times w_t^{(m)})$
END
$[\{\tilde{x}_t^{(n)}\}_{n=1}^{N}]$ = Replication $[\{x_t^{(m)}, N_t^{(m)}\}_{m=1}^{M}$

Combining (5) and (6), generates:

$$|N - N^*| =: \left| \sum_{i=1}^{M} (N \times W_i) - \sum_{i=1}^{M} (N \times W_i) \right| \leq \sum_{i=1}^{M} |N \times W_i - [N \times W_i]| <\approx \frac{M}{2} \tag{17}$$

The inequality (7) is then resolved based on the condition that $N^* \geq 0$, determining the final scope of N^* as mentioned.□

Experimental result

The preceding section talked about the rounding copy resampling algorithm's operation, which is applied if the physical memory exceeds 1.5 GB. This section talks about the proposed approach's (AMSSDR) experiment result. At present, several simulations were performed to examine the proposed algorithm's performance. The simulation of the experiment was performed using the parameters provided in Table 1, which were also utilised in other earlier resampling experiments [59–63].

In order to achieve a fair comparison, the resampling methods should undergo a comprehensive quantitative comparison based on the two varied physical memory requirements for the computing devices. This is done in order to determine if the proposed single distribution resampling (AMSSDR) will be able to function and last under a varying memory requirement. Two kinds of physical memory requirements will be utilised—4 GB (typical physical memory requirement) and 1.5 GB (low requirement for physical memory) (experiment reference is given in [64]). For the methods of resampling, the method that was utilised for simulation includes (when utilised as a point of comparison for the AMSSDR method): multinomial [62], systematic and stratified [60], RSR [61], residual [63], rounding-copy [54], and branch kill [59]. To obtain details for these algorithms, readers can refer to the pseudocodes provided in the tutorial (for these resampling methods, the MATLAB codes can be found at [65]).

The simulation makes use of various resampling methods. However, identical observation data is retained at every time step, and there will the same starting quantity of

Table 1 Parameter setting

Parameter setting	Value
Initial state (*x*)	0.1
Process noise covariance (*Q*)	10
Measurement noise covariance (*R*)	1
Simulation length (*tf*)	100
Number of particle in particle filter (*N*)	100

particles N for every filter. However, it will omit the selective resampling strategy that only implements resampling at certain steps. The filter estimate that is provided by all the particles' weighted mean is obtained after each filter is resampled so that it can directly reflect the effect that resampling has. The results of these filters utilising various resampling methods are plotted in the following figures.

As seen in the figures, others are an indication of all the resampling methods that the legend did not particularly specify. Generally, the true state and filter estimates are demonstrated when N_0 is given a value of 100 (demonstrated in Fig. 3) and the quantity of particles is determined using various resampling methods against the step (demonstrated in Fig. 4). To serve as an initial experiment, the single distribution resampling will be simulated under 4 GB of physical memory (typical physical memory requirement). It

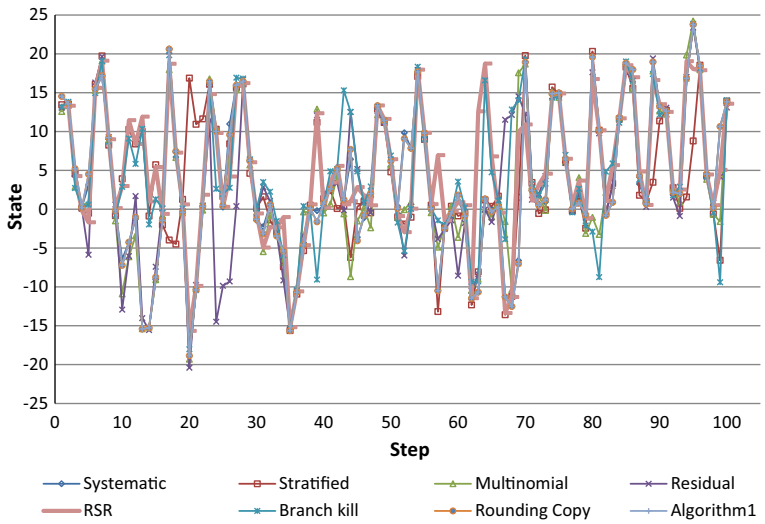

Fig. 3 Simulation of particle state for different types of single distribution resampling

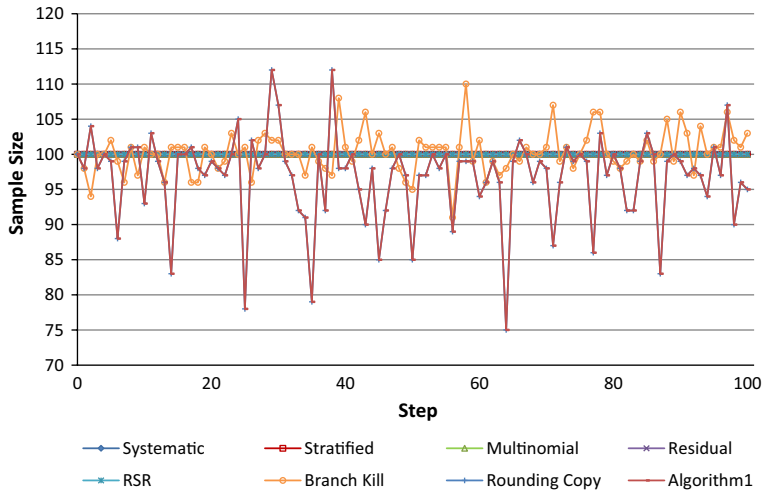

Fig. 4 Simulation of particle sample size for different types of single distribution resampling

demonstrates that even at every single run, they have almost identical resampling results (Fig. 3). This is in agreement with the qualitative study that was provided in the tutorial. Furthermore, it confirms that there is little difference among the unbiased algorithms. Figure 4 shows the plot of the fluctuation of the quantity of particles (sample size) against time. The figure demonstrates that the amount of particles vary within the range of 75–110 in rounding-copy, branch-kill, and AMSSDR. Furthermore, AMSSDR and rounding copy fluctuations were the same. It stays constant for other resampling methods when no mechanism is formulated to alter the amount of particles. Afterwards, a more in-depth discussion about single distribution resampling for 100 trials will be given in the next section. For these 100 trials, the average of the root mean square error (RMSE) results can be seen in "Root mean square error (RMSE)"; the sample size is given in "Sample size"; processing time is given in "Processing time"; and the proposed method's (AMSSDR) used memory is given in "Used memory".

Root mean square error (RMSE)

The preceding section talked about the experimental result. Moreover, it gave an outline of the experiment procedure that was used in this paper. This section will talk about the RMSE that was gathered from the simulated resampling algorithms. The main findings are summarised using these three points: First, resampling is vital in this filtering model, as shown by the sequence important sampling (SIS) filter's evidently low accuracy. Thus, the SIS filter will not be discussed further. Second, in terms of RMSE, all unbiased resampling methods will provide equivalent estimation accuracy (Figs. 5, 6), especially when the number of particles exceeds 100. Lastly, the value of the RMSE of Algorithm 1 (known as AMSSDR) is similar to the value for the rounding copy resampling when it is employed in a computing device that has a typical memory requirement. On the other hand, RMSE is similar to systematic resampling when it is employed in a computing device with low memory requirement.

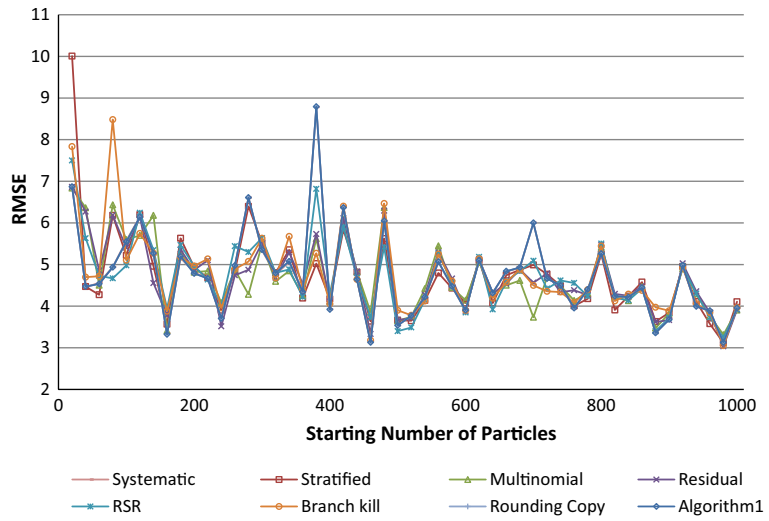

Fig. 5 Simulation of RMSE with typical requirement memory platform; the AMSSDR and rounding copy resampling have a common type of RMSE

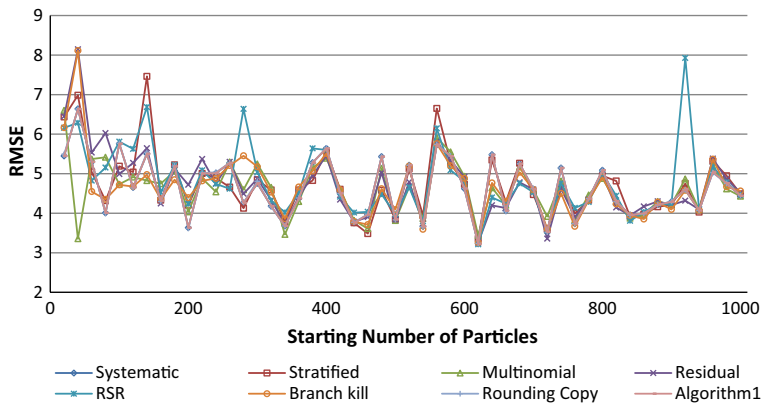

Fig. 6 Simulation of RMSE with low requirement memory platform; The AMSSDR and systematic resampling have same common of RMSE

Sample size

The preceding section deliberated RMSE gained from a simulated resampling technique. This current section considers the sample size of the simulated resampling technique.

As shown in Fig. 7 as well as Fig. 8, the following deductions can be ascertained. It is demonstrated that the branch-kill, AMSSDR, and rounding-copy obtain the nearest number of units to N_0 in descending order. Compared to that, the branch kill is the most stable. Generally, the rounding-copy obtained a somewhat smaller number of units than the reference, in spite of its mean being the same as the reference in theory [54]. It is inferred that this is because of the MATLAB software that uses limited-precision storage to shorten the float number. Furthermore, the sample size of method 1 (known as AMSSDR) is same as the rounding copy resampling technique when implemented in a standard memory requirement computing device, whereas the sample size is same as that of the systematic resampling technique, when implemented in a computing device having low memory requirement.

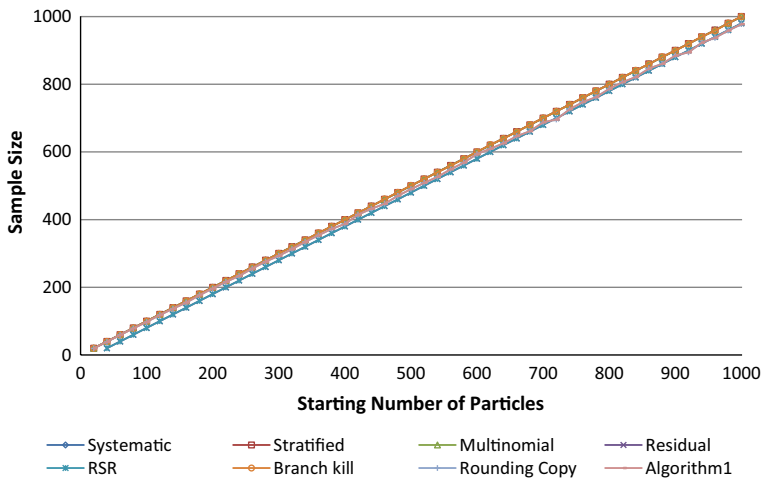

Fig. 7 Simulation with typical requirement memory platform; the AMSSDR and rounding copy resampling have a common type of sample size

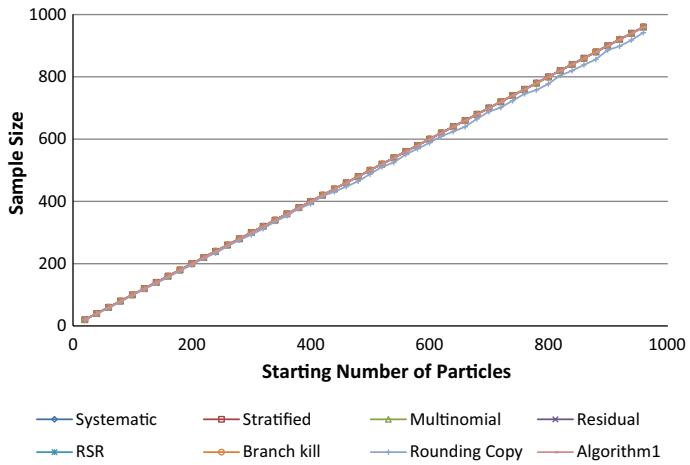

Fig. 8 Simulation with low requirement memory platform; the AMSSDR and systematic resampling have a common type of sample size

Processing time

The preceding section considered the sample size of the modelled resampling algorithm. This present section reviews the time of processing of the simulated resampling technique. For the testing period of 100 steps, the average processing time of various resampling methods alongside the starting number of units N_0 from 20 to 500 in 100 trials is provided in Fig. 9 as well as Fig. 10. It should be observed that the speed of computing depends on the hardware platform and the technology used for programming, for which all resampling techniques have been accelerated in the nearest equivalent approach possible. Given this requirement, the results provided in Fig. 6 signify the following basic conclusion. When containing the uniform starting number of units, different resampling techniques have significantly different speeds of computing, and do not, at all times, rank in the same range. When is too small, by increasing the value of N, the computing time necessary increases more notably than for others.

This is cautiously considered when selecting a resampling method. Amongst these, the algorithm 1 resampling (AMSSDR) is the most time-consuming, due to the requirement

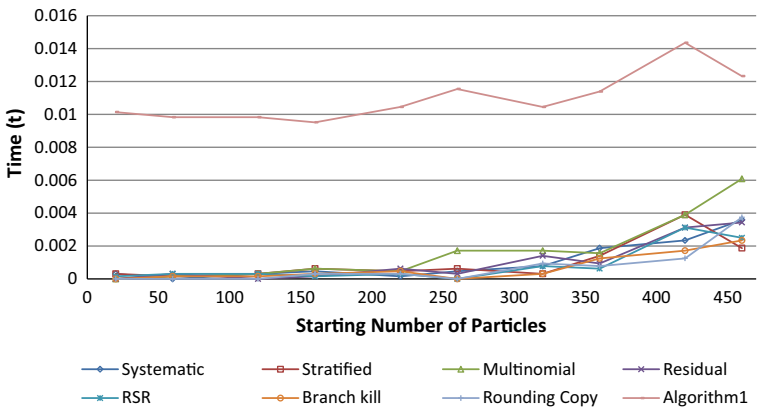

Fig. 9 Simulation of resampling with typical requirement memory platform

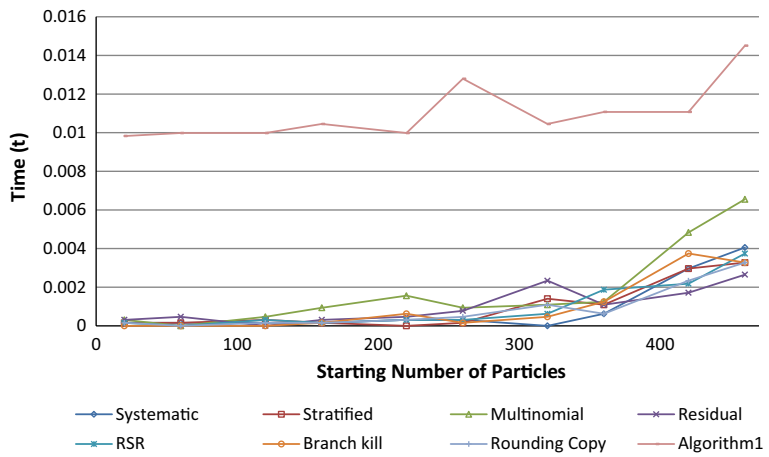

Fig. 10 Simulation of resampling with low requirement memory platform

to switch suitable resampling depending on memory requirement, which uses too much computing power. Thus, in general, the simulation outcomes demonstrate that: First of all, the resampling techniques rarely produce greatly different results if they fulfil the unbiasedness (even asymptotically) criteria, preserving a fixed number of particles and uniformly weight resampled particles. Another thing is, if these constraints are removed, special advantages may be obtained, for instance, adaptively adjusting the amount of particles according to the requirement of the system, and disregarding unbiasedness so as to preserve particle variety, alleviate impoverishment, or facilitate parallel processing. Nevertheless, these new benefits come at the price of added computational requirements (for instance, due to the complicated algorithm design), and are usually model specific. Third, various resampling methods may have considerably different computational speeds, based on the amount of particles and the model of the problem. In general, deterministic and single sampling techniques are computationally more rapid than random and complex sampling methods, and so are more appropriate for parallel implementation.

Used memory

The preceding section reviewed the sample size of the simulated resampling technique. This present section reviews the used memory of the AMSSDR technique. For the *j* cycle duration of hundred (100) steps (as per Fig. 11 as well as Fig. 12), the amount of particles against the *j* cycle changes, when implemented in a standard memory computing device, for instance, becoming constant when executed in a computing device having low memory. This is because of the operation of the suggested approach (AMSSDR) which is similar to the rounding copy resampling technique when implemented in a standard memory computing device, whereas the operation of the suggested approach will be similar to systematic resampling. This creates the pattern of all the performances of the suggested approach (AMSSDR). In the meantime, for the *j* cycle duration of 100 steps (as per the Fig. 13 as well as Fig. 14), the quantity of particles against the *j* cycle changes between 0 and 32 bytes, when it is executed in a typical memory computing device, whereas it becomes constant at 8 bytes, when executed in a computing device

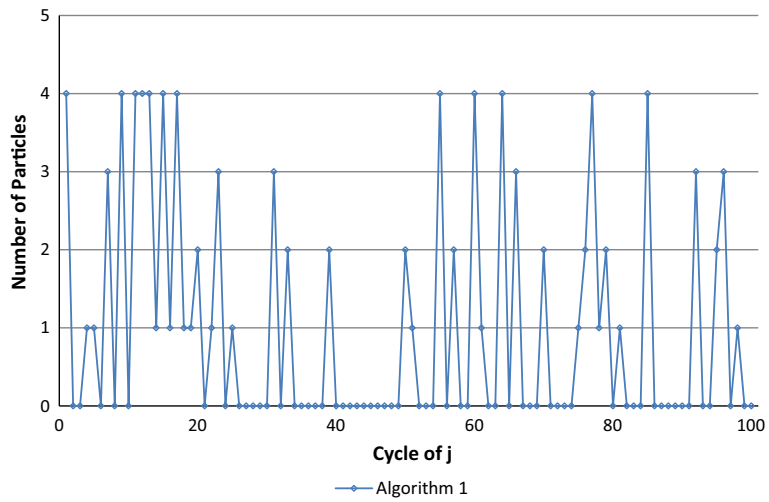

Fig. 11 Simulation number of particles with typical requirement memory platform

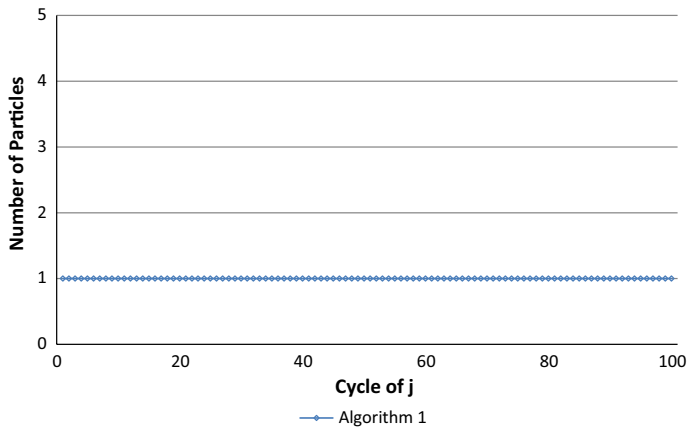

Fig. 12 Simulation number of particleswith low requirement memory platform

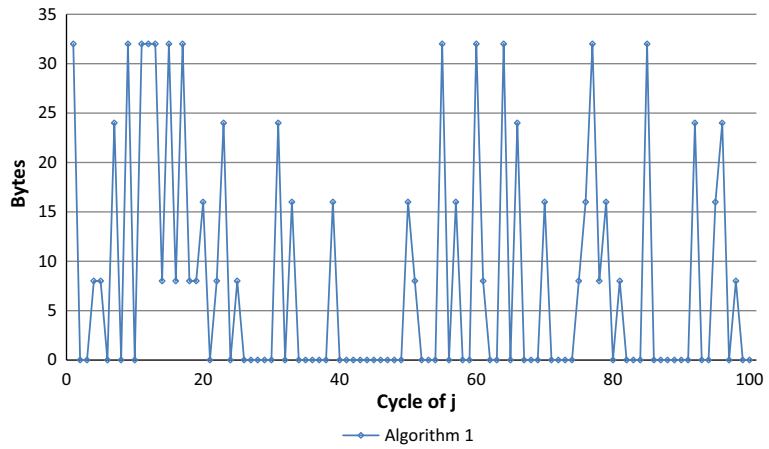

Fig. 13 Simulation with typical requirement memory platform (left) and low requirement memory platform (right)

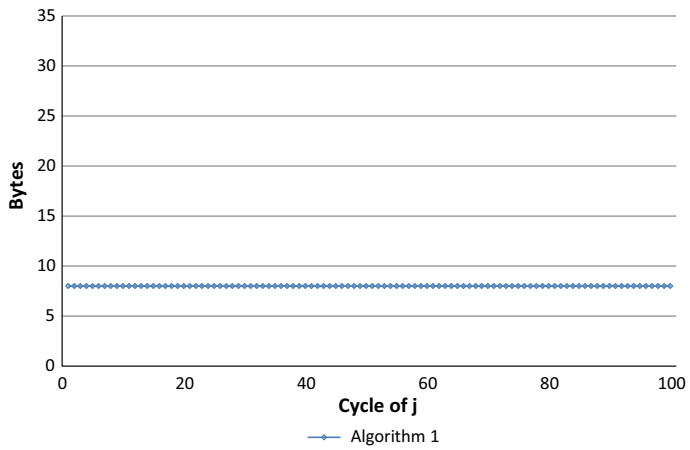

Fig. 14 Simulation with typical requirement memory platform (left) and low requirement memory platform (right)

having low memory. This is because of the number of particles changing when implemented into a standard memory computing device, whereas the number of particles is fixed and constant when executed into a low memory computing device.

Conclusions and future implementations

The preceding section considered the used memory of the AMSSDR technique. This present section reviews the conclusion of the suggested method (AMSSDR). Restricting the use of single distribution resampling in case of the memory of specific computing devices causes difficulties for the developer, because of the additional time and effort needed to develop a particle filter. Hence, a new sequential resampling technique is needed, one with an amount which requires memory size depending on the memory requirement of computing devices. In this research, a new single distribution resampling technique has been created, known as AMSSDR, which is based on the combination of traditional resampling technique and traditional variation resampling technique in a resampling architecture. The technique will switch the resampling algorithm which is based on memory in a computer. The execution of this algorithm will facilitate developers to more effortlessly develop a particle filter, with no need to give a big degree of consideration to memory usage in a computing device when including different particle filter development. Initially, in the operational process, the AMSSDR selector will be utilised to select an appropriate resampling algorithm (for instance, systematic resampling technique or rounding copy resampling technique), as per the physical memory available in present computing devices. Following that, the outcomes show that, if systematic resampling is chosen, the resampling will take sample for each particle for each j cycle, while if the rounding copy resampling is selected, the resampling will take samples for more than one unit of each j cycle. Hence, this demonstrates that the suggested method (AMSSDR) is capable of switching and resampling algorithms in various physical memory requirements. The authors of this paper wish to extend this work gradually by implementing their suggested method in a number of different promising applications (for instance, in medical software [66] or real-time locator systems [67].

Authors' contributions
WMYWB has contributed for acquisition of data, analysis, interpretation of data, and drafting of the manuscript. MMM has served as scientific advisors in study conception, design and for critical revision. RZRMR has critically reviewed the study proposal. MS and AFY has served as reviewer and giving many comment to in order to improve this manuscript. All authors read and approved the final manuscript.

Author details
[1] Faculty of Computing, Universiti Teknologi Malaysia, Johor Bahru, Johor, Malaysia. [2] Faculty of Information and Communication Technology, Universiti Teknikal Malaysia, Melaka, Malaysia.

Acknowledgements
Not applicable.

Competing interests
The authors declare that they have no competing interests.

Availability of supporting data
All supporting data files are available and can be accessed at https://sites.google.com/site/tianchengli85/home.

Consent for publication
Both authors are offering the consent to publish the research.

Ethical approval and consent to partake
Ethical approval has been acquired from Universiti Teknologi Malaysia, Malaysia.

Funding information
This paper has been funded by Research University Grant (Title: Prototype of Android 6.0 Indoor Tracking Systems based on Intelligent Fusion WiFi/IMU Signals, Cost Centre No: Q.J130000.2528.13H60).

Publisher's Note

Received: 24 July 2017 Accepted: 26 September 2017
Published online: 18 October 2017

References

1. Liu JS, Chen R, Logvinenko T. A theoretical framework for sequential importance sampling with resampling. In: Doucet A, de Freitas N, Gordon N, editors. Sequential monte carlo methods in practice. New York: Springer; 2001. p. 225–46.
2. Douc R, Cappe O. Comparison of resampling schemes for particle filtering, ISPA 2005. In: Proceedings of the 4th international symposium on image and signal processing and analysis. 2005. p. 64–69.
3. Bolić M, Djurić PM, Hong S. Resampling algorithms for particle filters: a computational complexity perspective. EURASIP J Appl Signal Process. 2004;2004:2267–77.
4. Hol JD, Schon TB, Gustafsson F. On resampling algorithms for particle filters, 2006. In: IEEE nonlinear statistical signal processing workshop. 2006, p. 79–82.
5. Zhou H, Deng Z, Xia Y, Fu M. A new sampling method in particle filter based on Pearson correlation coefficient. Neurocomputing. 2016;216:208–15.
6. Tulsyan A, Bhushan Gopaluni R, Khare SR. Particle filtering without tears: a primer for beginners. Comput Chem Eng. 2016;95:130–45.
7. Bejuri WMYW, Mohamad MM, Zahilah R. Offline beacon selection-based RSSI fingerprinting for location-aware shopping assistance: a preliminary result. In: New trends in intelligent information and database systems. Berlin: Springer; 2015. p. 303–312.
8. Bejuri WMYW, Mohamad MM. Optimisation of emergency rescue location (ERL) using KLD resampling: an initial proposal. Int J U E Serv Sci Technol. 2016;9(2):249–62.
9. Vázquez MA, Míguez J. A robust scheme for distributed particle filtering in wireless sensors networks. Signal Process. 2017;131:190–201.
10. Míguez J, Vázquez MA. A proof of uniform convergence over time for a distributed particle filter. Signal Process. 2016;122:152–63.
11. Bejuri WMYW, Mohamad MM, Radzi RZRM, Salleh M, Yusof AF., Adaptive memory size based fuzzy control for mobile pedestrian navigation. In: Lecture notes on data engineering and communications technologies, vol. 5. Cham: Springer. 2017. p. 132–140.
12. Bejuri WMYW, Mohamad MM, Radzi RZRM, Salleh M, Yusof AF. Adaptive resampling for emergency rescue location: an initial ioncept. In: Lecture notes in ectrical engineering. vol. 425, Singapore: Springer. 2017. p. 269–273.
13. Manoli G, et al. An iterative particle filter approach for coupled hydro-geophysical inversion of a controlled infiltration experiment. J Comput Phys. 2015;283:37–51.
14. Da Silva LA, et al. Comparison of geophysical patterns in the southern hemisphere mid-latitude region. Adv Space Res. 2016;58(10):2090–103.

15. Bejuri WMYW, Mohamad MM, Zahilah R. A proposal of emergency rescue location (ERL) using optimization of inertial measurement unit (IMU) based pedestrian simultaneously localization and mapping (SLAM). Int J Smart Home. 2015;9(12):9–22.
16. Bejuri WMYW, Mohamad MM, Zahilah R. Emergency rescue localization (ERL) using GPS, wireless LAN and camera. Int J Softw Eng Appl. 2015;9(9):217–32.
17. Bejuri WMYW, Mohamad MM, Zahilah R. Optimization of Rao-Blackwellized particle filter in activity pedestrian simultaneously localization and mapping (SLAM): an initial proposal. Int J Secur Appl. 2015;9(11):377–90.
18. Wang B, Yu L, Deng Z, Fu M. A particle filter-based matching algorithm with gravity sample vector for underwater gravity aided navigation. IEEEASME Trans Mechatron. 2016;21(3):1399–408.
19. Teixeira FC, Quintas J, Maurya P, Pascoal A. Robust particle filter formulations with application to terrain-aided navigation. Int J Adapt Control Signal Process. 2017;31(4):608–51.
20. Bejuri WMYW, Mohamad MM, Sapri M, Rosly MA. Investigation of color constancy for ubiquitous wireless LAN/camera positioning: an initial outcome. Int J Adv Comput Technol IJACT. 2012;4(7):269–80.
21. Ploumpis S, Amanatiadis A, Gasteratos A. A stereo matching approach based on particle filters and scattered control landmarks. Image Vis Comput. 2015;38:13–23.
22. Schwiegelshohn F, Ossovski E, Hübner M. A resampling method for parallel particle filter architectures. Microprocess Microsyst. 2016;47:314–20.
23. Bejuri WMYW, Mohamad MM. Performance analysis of grey-world-based feature detection and matching for mobile positioning systems. Sens Imaging. 2014;15(1):95.
24. Bejuri WMYW, Mohamad MM, Sapri M, Rosly MA. Performance evaluation of mobile U-navigation based on GPS/WLAN hybridization. J Converg Inf Technol JCIT. 2012;7(12):235–46.
25. Bejuri WMYW, Mohamad MM, Sapri M, Rahim MSM, Chaudry JA. Performance evaluation of spatial correlation-based feature detection and matching for automated wheelchair navigation system. Int J Intell Transp Syst Res. 2014;12(1):9–19.
26. Zhang Y, Ji H, Hu Q. A box-particle implementation of standard PHD filter for extended target tracking. Inf. Fusion. 2017;34:55–69.
27. Ibarguren A, Maurtua I, Pérez MA, Sierra B. Multiple target tracking based on particle filtering for safety in industrial robotic cells. Robot Auton Syst. 2015;72:105–13.
28. Bejuri WMYW, Mohamad MM, Sapri M. Ubiquitous positioning: a taxonomy for location determination on mobile navigation system. Signal Image Process Int J SIPIJ. 2011;2(1):24–34.
29. Bejuri WMYW, Saidin WMNWM, Mohamad MM, Sapri M, Lim KS. Ubiquitous positioning: integrated GPS/wireless LAN positioning for wheelchair navigation system. In: Intelligent information and database systems. vol. 7802. 2013. p. 394–403.
30. Bejuri WMYW, Mohamad MM, Sapri M, Rosly MA. Ubiquitous WLAN/camera positioning using inverse intensity chromaticity space-based Feature detection and matching: a preliminary result. In: International conference on man–machine systems 2012 (ICOMMS 2012), 2012.
31. Bejuri WMYW, Mohamad MM. Wireless LAN/FM radio-based robust mobile indoor positioning: an initial outcome. Int J Softw Eng Appl. 2014;8(2):313–24.
32. Gordon NJ, Salmond DJ, Smith AF. Novel approach to nonlinear/non-Gaussian Bayesian state estimation. IEE Proc F-Radar Signal Process. 1993;140:107–13.
33. Rubin DB. The calculation of posterior distributions by data augmentation: comment: a noniterative sampling/importance resampling alternative to the data augmentation algorithm for creating a few imputations when fractions of missing information are modest: the SIR algorithm. J Am Stat Assoc. 1987;82(398):543–6.
34. Juang C-F. A hybrid of genetic algorithm and particle swarm optimization for recurrent network design. IEEE Trans. Syst. Man Cybern. Part B Cybern. 2004;34(2):997–1006.
35. Moradkhani H, Hsu KL, Gupta H, Sorooshian S. Uncertainty assessment of hydrologic model states and parameters: sequential data assimilation using the particle filter. Water Resour Res. 2005;41(5):1–17.
36. Li T, Sun S, Sattar TP, Corchado JM. Fight sample degeneracy and impoverishment in particle filters: a review of intelligent approaches. Expert Syst Appl. 2014;41(8):3944–54.
37. Whitacre JM, Atamas SP. Degeneracy allows for both apparent homogeneity and diversification in populations. Biosystems. 2012;110(1):34–42.
38. Ingrassia S, Rocci R. Degeneracy of the EM algorithm for the MLE of multivariate Gaussian mixtures and dynamic constraints. Comput Stat Data Anal. 2011;55(4):1715–25.
39. Ginsbourger D, Roustant O, Durrande N. On degeneracy and invariances of random fields paths with applications in Gaussian process modelling. J Stat Plan Inference. 2016;170:117–28.
40. Li T, Sattar TP, Sun S. Deterministic resampling: unbiased sampling to avoid sample impoverishment in particle filters. Signal Process. 2012;92(7):1637–45.
41. Sbarufatti C, Corbetta M, Manes A, Giglio M. Sequential Monte-Carlo sampling based on a committee of artificial neural networks for posterior state estimation and residual lifetime prediction. Int J Fatigue. 2016;83:10–23.
42. Sharifian MS, Rahimi A, Pariz N. Classifying the weights of particle filters in nonlinear systems. Commun Nonlinear Sci Numer Simul. 2016;31(1):69–75.
43. Tagade P, et al. Recursive bayesian filtering framework for lithium-ion cell state estimation. J Power Sources. 2016;306:274–88.
44. Tian Q, Pan Y, Salcic Z, Huan R. DART: distributed particle filter algorithm with resampling tree for ultimate real-time capability. J Signal Process Syst. 2017;88(1):29–42.
45. Yin S, Zhu X, Qiu J, Gao H. State estimation in nonlinear system using sequential evolutionary filter. IEEE Trans Ind Electron. 2016;63(6):3786–94.
46. Wu J, Chen Y, Zhou S, Li X. Online steady-state detection for process control using multiple change-point models and particle filters. IEEE Trans Autom Sci Eng. 2016;13(2):688–700.
47. Beadle ER, Djuric PM. A fast-weighted Bayesian bootstrap filter for nonlinear model state estimation. IEEE Trans Aerosp Electron Syst. 1997;33(1):338–43.

48. Bolić M, Djurić PM, Hong S. Resampling algorithms for particle filters: a computational complexity perspective. EURASIP J Appl Signal Process. 2004;2004:2267–77.
49. Efron B, Tibshirani RJ. An introduction to the bootstrap. Boca Raton: CRC press; 1994.
50. Kitagawa G. Monte Carlo filter and smoother for non-Gaussian nonlinear state space models. J Comput Graph Stat. 1996;5(1):1–25.
51. Carpenter J, Clifford P, Fearnhead P. Improved particle filter for nonlinear problems. IEE Proc Radar Sonar Navig. 1999;146(1):2–7.
52. Fearnhead P. Sequential Monte Carlo methods in filter theory. University of Oxford: Oxford; 1998.
53. Budhiraja A, Chen L, Lee C. A survey of numerical methods for nonlinear filtering problems. Phys Nonlinear Phenom. 2007;230(1):27–36.
54. Li T, Sattar TP, Tang D. A fast resampling scheme for particle filters. In: Signal processing (CIWSP 2013), 2013 constantinides international workshop on 2013. 2013. p. 1–4.
55. Statistics—Facebook. http://goo.gl/qwyFHe. Accessed 13 Nov 2016.
56. Statistics—YouTube. https://www.youtube.com/yt/press/statistics.html. Accessed 13 Nov 2016.
57. Dong X, Muralimanohar N, Jouppi N, Kaufmann R, Xie Y. Leveraging 3D PCRAM technologies to reduce checkpoint overhead for future exascale systems. In: Proceedings of the conference on high performance computing networking, storage and analysis. 2009. p. 57.
58. Li T, Villarrubia G, Sun S, Corchado JM, Bajo J. Resampling methods for particle filtering: identical distribution, a new method, and comparable study. Front Inf Technol Electron Eng. 2015;16(11):969–84.
59. Crisan D, Lyons T. A particle approximation of the solution of the Kushner–Stratonovitch equation. Probab Theory Relat Fields. 1999;115(4):549–78.
60. Kitagawa G. Monte Carlo filter and smoother for non-Gaussian nonlinear state space models. J Comput Graph Stat. 1996;5(1):1–25.
61. Bolic M, Djuric PM, Hong S. New resampling algorithms for particle filters in Acoustics, speech, and signal processing, 2003. In: Proceedings (ICASSP'03) 2003 IEEE international conference on 2003. vol. 2. p. II–589.
62. Gordon NJ, Salmond DJ, Smith AF. Novel approach to nonlinear/non-Gaussian Bayesian state estimation. IEE Proc F Radar Signal Process. 1993;140:107–13.
63. Liu JS, Chen R. Sequential Monte Carlo methods for dynamic systems. J Am Stat Assoc. 1998;93(443):1032–44.
64. Grisetti G, Tipaldi GD, Stachniss C, Burgard W, Nardi D. Fast and accurate SLAM with Rao-Blackwellized particle filters. Robot Auton Syst. 2007;55(1):30–8.
65. Resampling codes for PF. In: TC Li. https://sites.google.com/site/tianchengli85/matlab-codes/resampling-methods. Accessed: 17 Nov 2016.
66. Du G, Zhang P, Liu X. Markerless human-manipulator interface using leap motion with interval Kalman filter and improved particle filter. IEEE Trans Ind Inform. 2016;12(2):694–704.
67. Shane DD, White BJ, Larson RL, Amrine DE, Kramer JL. Probabilities of cattle participating in eating and drinking behavior when located at feeding and watering locations by a real time location system. Comput Electron Agric. 2016;127:460–6.